KONRAD LORENZ
FUNDACIÓN UNIVERSITARIA

Electiva I: Backend para Web, Móvil e IoT

Tema 2

Profesor: Jorge Iván Andrés Contreras Pereira

Agenda

Agenda

- ¿Qué es REST?
- Ejemplos
- JSON

REST

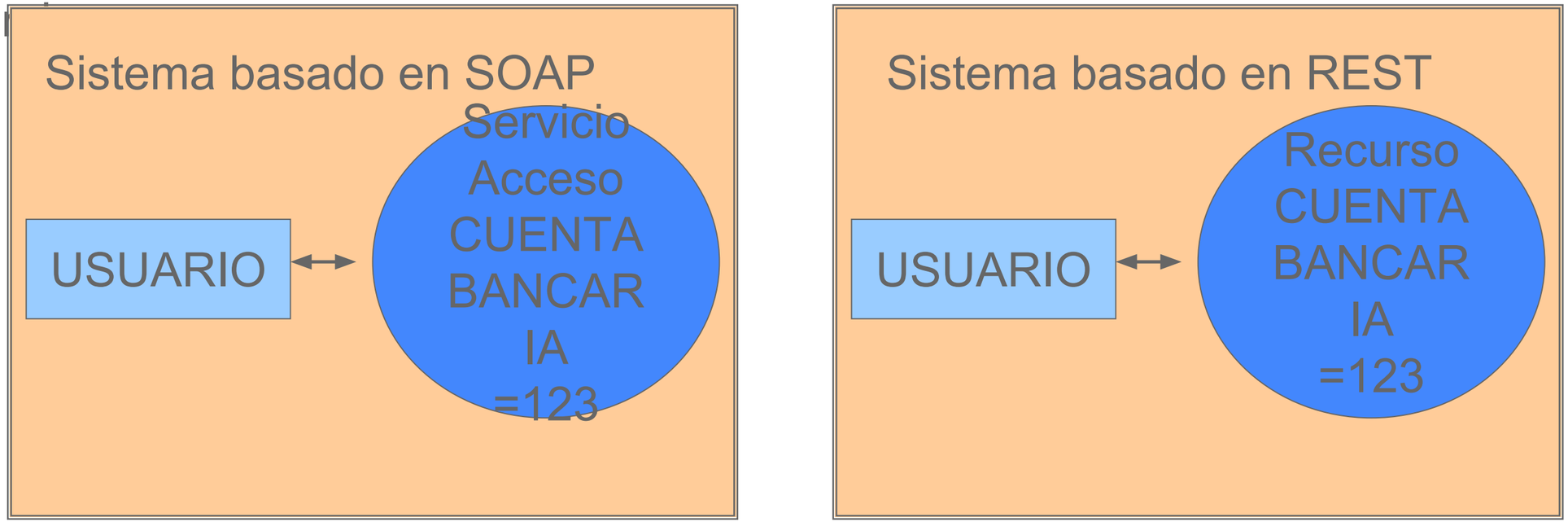
- Origen: Fielding, Roy T. “Architectural Styles and the Design of Network-based Software Architectures.” Tesis Doctoral, Universidad de California, 2000.
- Describe un estilo de arquitectura que utilizar como modelo en los servicios de computación Web.
- Estilo de arquitectura: Conjunto coordinado de **restricciones** que controlan el funcionamiento y características de los elementos de la arquitectura y permite las relaciones de unos con otros.
- Describe **cómo debería comportarse la Web**
- **NO** es un estándar

¿Por qué ha triunfado la Web?

- Escalabilidad en interacciones entre componentes
- Generalidad en las interfaces
- Desarrollo independiente de componentes
- Existencia de componentes intermediarios (proxys)

Principios

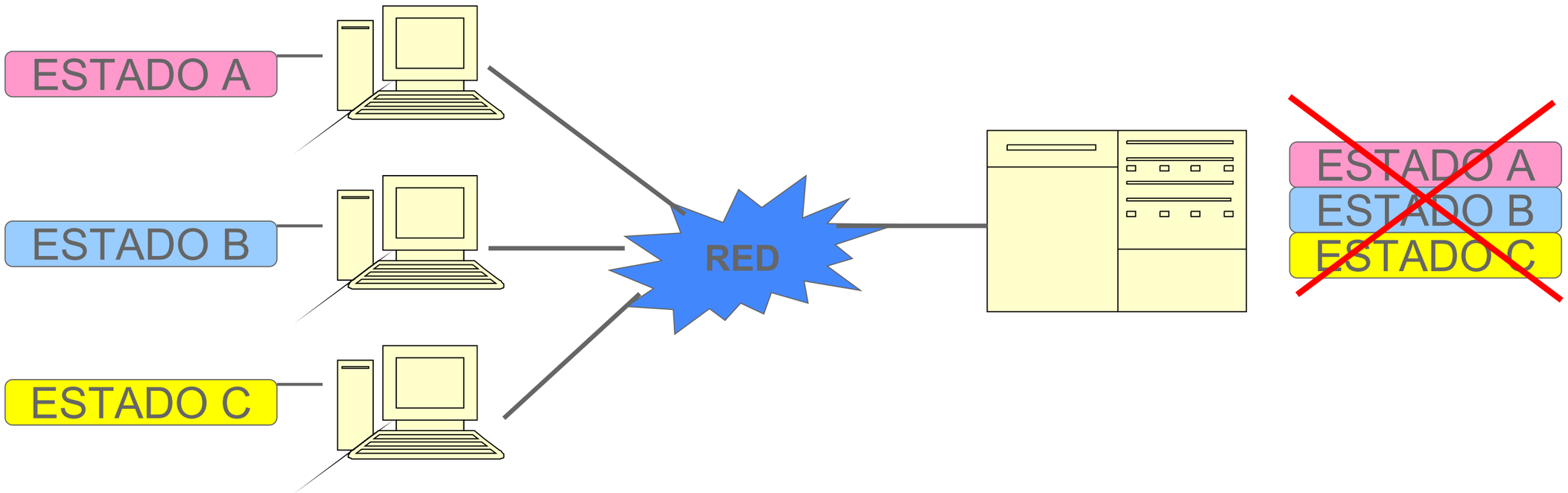
- El estado y la funcionalidad de las aplicaciones se divide en **recursos**
  - REST es orientado a recursos y no a métodos
  - No se accede directamente a los recursos, sino a representaciones de los



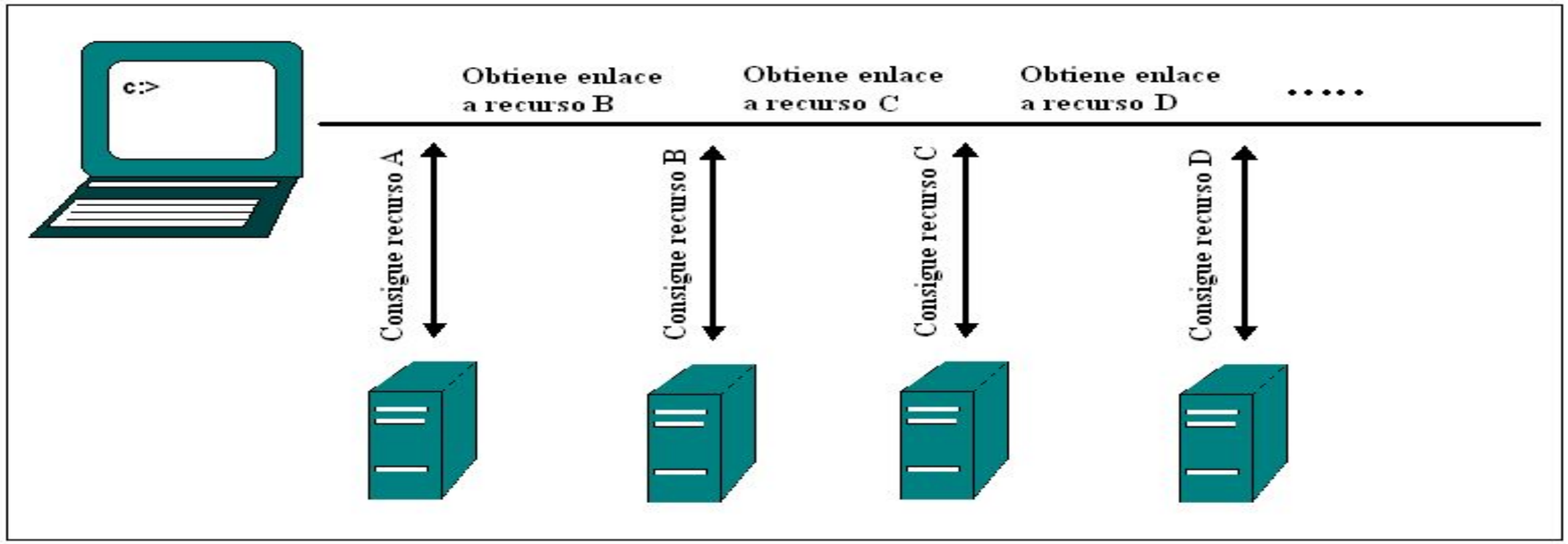
- Todo recurso es identificado de **forma única global** mediante una sintaxis universal. Como en HTTP los recursos se identifican mediante URIs *(Uniform Resource Identifier)*.
  - **Conjunto potencialmente infinito de recursos.**
- Todos los recursos comparten un **interfaz uniforme** formado por:
  - **Conjunto de operaciones** limitado para transferencia de estado
    - En HTTP GET, PUT, POST, DELETE
  - **Conjunto** limitado **de tipos de contenidos**
    - En HTTP se identifican mediante tipos MIME: XML , HTML...

MÉTODO	FUNCIÓN
GET	Solicitar recurso
POST	Crear recurso nuevo
PUT	Actualizar o modificar recurso
DELETE	Borrar recurso

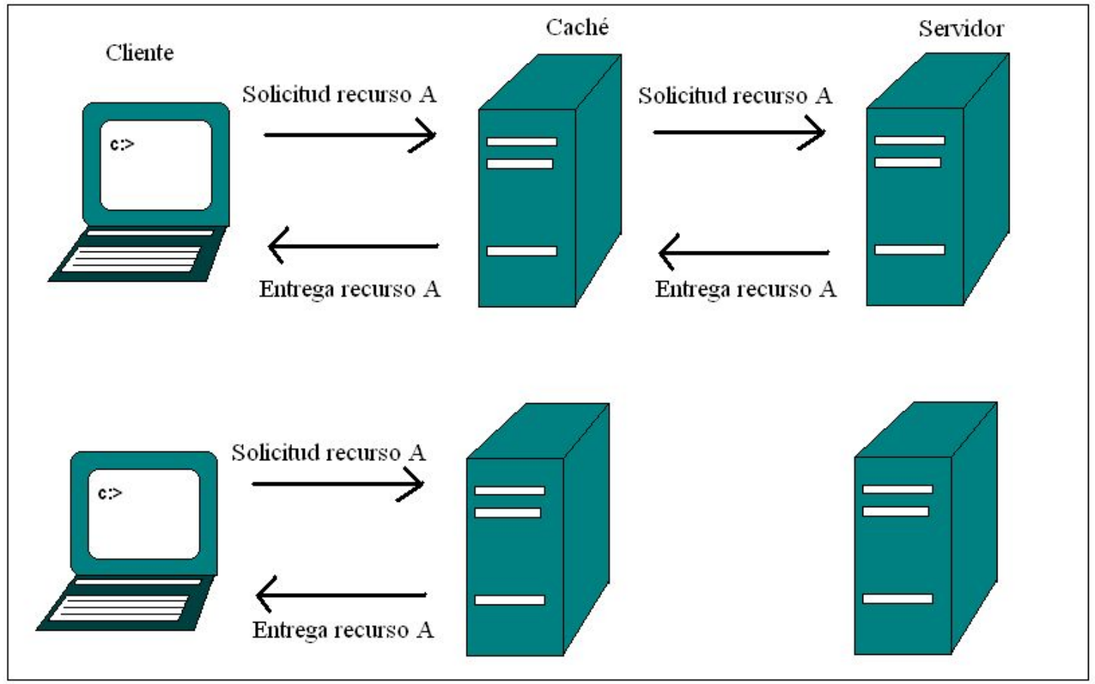
- Un protocolo cliente/servidor, sin estado y basado en capas
- Cada mensaje contiene la información necesaria para comprender la petición (mensajes autocontenidos, como HTTP)



- **Uso de hipermedios**, tanto para la información de la aplicación como para las transiciones de estado de la aplicación.
- A través de sucesivas peticiones de recursos cambia el estado de la aplicación.



- **Promueve mecanismos caché y sistemas intermedios**



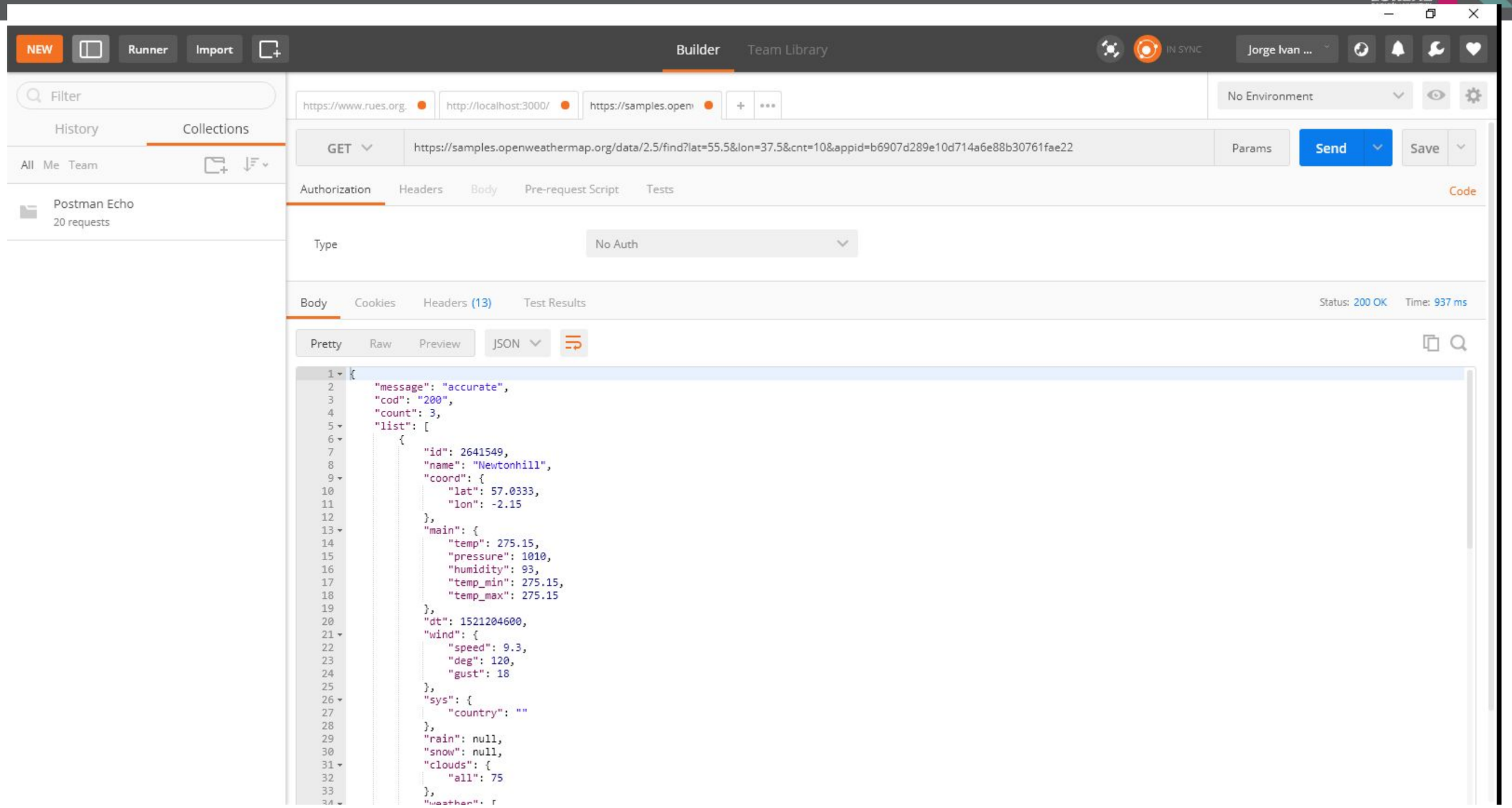
Ventajas

- Mejora el tiempo de respuesta gracias al mecanismo Caché y los mensajes auto-descriptivos.
- Disminución de carga en servidor
- Mayor escalabilidad al no requerir mantenimiento de estado en el servidor
- Facilita desarrollo de clientes (menor dependencia del servidor).
- Mayor estabilidad frente a futuros cambios
  - Permite evolución independiente de los tipos de documentos al procesar éstos en el cliente.

Diferencias

SOAP	REST
Orientado a RPC	Orientado a recursos
Servidor almacena parte del estado	El estado se mantiene sólo en el cliente, y no se permiten las sesiones
Usa HTTP como túnel para el paso de mensajes	Propone HTTP como nivel de aplicación

Ejemplo


NEW
Runner
Import
Builder
Team Library
IN SYNC
Jorge Ivan ...
Filter
History
Collections
All Me Team
Postman Echo
20 requests
https://www.rues.org.
http://localhost:3000/
https://samples.open
No Environment
GET
https://samples.openweathermap.org/data/2.5/find?lat=55.5&lon=37.5&cnt=10&appid=b6907d289e10d714a6e88b30761fae22
Params
Send
Save
Authorization
Headers
Body
Pre-request Script
Tests
Code
Type
No Auth
Body
Cookies
Headers (13)
Test Results
Status: 200 OK
Time: 937 ms
Pretty
Raw
Preview
JSON
{
"message": "accurate",
"cod": "200",
"count": 3,
"list": [
{
"id": 2641549,
"name": "Newtonhill",
"coord": {
"lat": 57.0333,
"lon": -2.15
},
"main": {
"temp": 275.15,
"pressure": 1010,
"humidity": 93,
"temp_min": 275.15,
"temp_max": 275.15
},
"dt": 1521204600,
"wind": {
"speed": 9.3,
"deg": 120,
"gust": 18
},
"sys": {
"country": ""
},
"rain": null,
"snow": null,
"clouds": {
"all": 75
},

Críticas

- SOAP no es transparente, apuesta por el encapsulamiento
- SOAP no dispone de un sistema de direccionamiento global
- SOAP puede derivar en agujeros de seguridad
- SOAP no aprovecha muchas de las ventajas de HTTP al usarlo solamente como túnel
- SOAP no puede hacer uso de los mecanismos Caché

- REST es poco flexible
- REST no está preparado para albergar Servicios Web de gran complejidad como las aplicaciones B2B
- REST tiene grandes problemas de seguridad al no soportar el concepto de sesión

¿Cuándo usar?

- Adecuado para grandes cantidades de información pública para grupos desconocidos de usuarios
- No adecuado para sistemas complejos cerrados

JSON

JSON

JSON, acrónimo de "**J**ava**S**cript **O**bject **N**otation", es un formato ligero para el intercambio de datos. **JSON** es un subconjunto de la notación literal de objetos de Javascript pero no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XMLLa simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de **JSON** sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador semántico de JSON.

En Javascript, JSON puede ser analizado trivialmente usando el procedimiento eval(), lo cual ha sido fundamental para la aceptación de JSON por parte de la comunidad de desarrolladores Ajax, debido a la ubicuidad de Javascript en casi cualquier navegador Web.

JSON

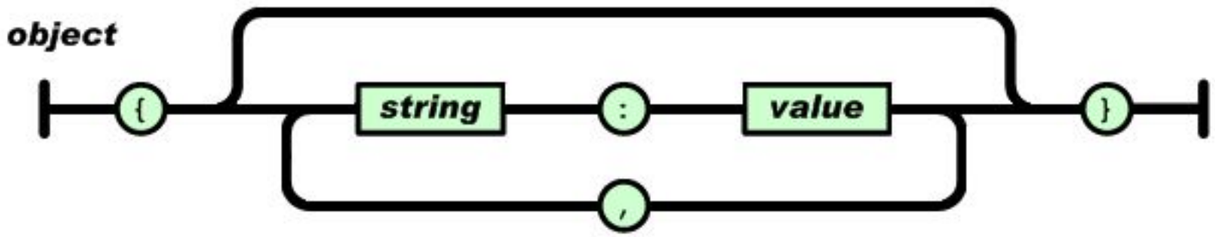
JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

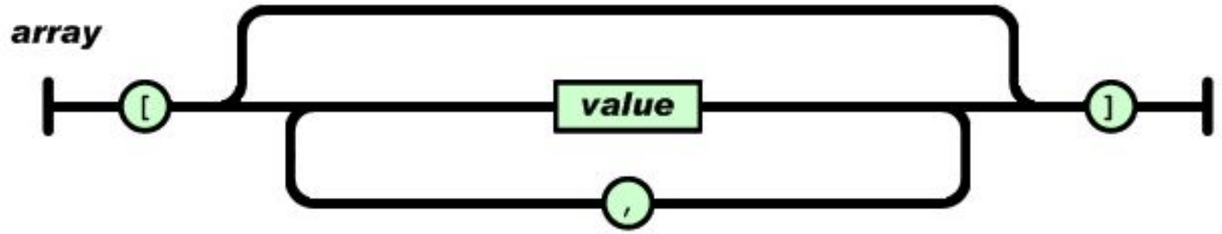
JSON

Un *objeto* es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma).



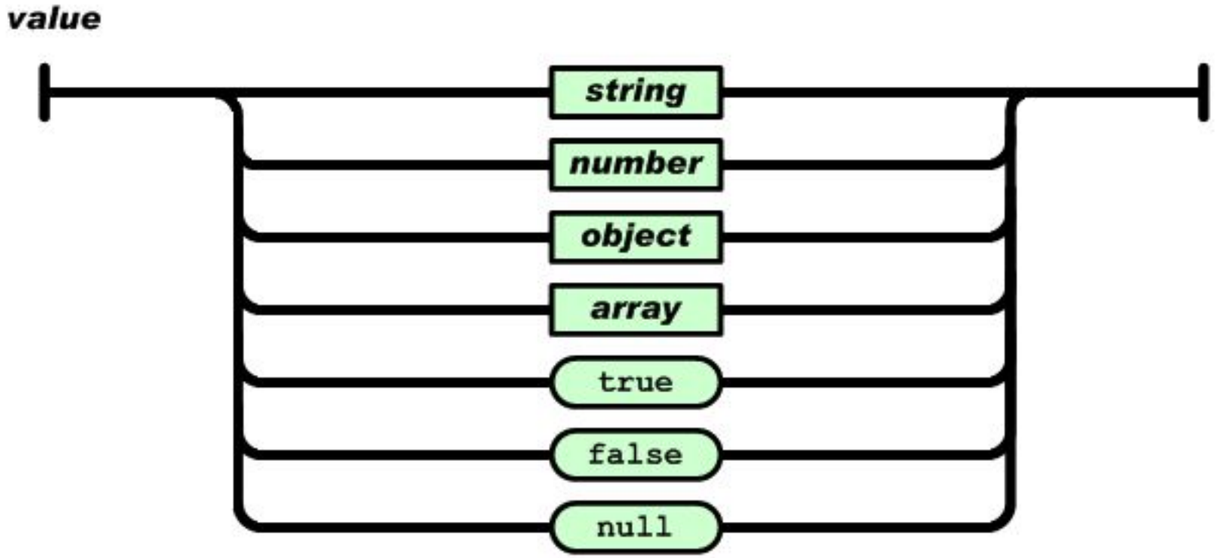
JSON

Un *array* es una colección de valores. Un array comienza con [(corchete izquierdo) y termina con] (corchete derecho). Los valores se separan por , (coma).



JSON

Un *valor* puede ser una *cadena de caracteres* con comillas dobles, o un *número*, o true o false o null, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse



JSON – Equivalencia con XML

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON – Enlaces de Interés

- http://es.wikipedia.org/wiki/JSON
- http://json.org/
- http://json.org/json-es.html

Trabajar con JSON desde ASP

- http://www.webdevbros.net/2007/04/26/generate-json-from-asp-datatypes/
- http://www.webdevbros.net/2007/08/21/json-utility-class-13-released/

JSON – Formato datos

Formato

{“objeto": [{“campo1": valor1,“campo2": “valor2",..}]}

Ejemplo

{"personas": [{"idpersona": 3,"nombre": "ALFONSO","apellidos": "BENAVENT VICTORIA","email": "ABenavent@ua.es"}]}

Gracias